Discovering Sequential Disease Patterns in Medical Databases Using FreeSpan Mining Approach

Oviliani Yenty Yuliana, Silvia Rostianingsih, and Gregorius Satia Budhi
Informatics Engineering Department
Industrial Technology Faculty, Petra Christian University
Email: ovi@petra.ac.id, silvia@petra.ac.id, greg@petra.ac.id

***Abstract*— In this work we employ FreeSpan, one of the sequential pattern mining methods, to discover sequential disease patterns in sequential database. For our study, we use a medical database which was built as a transactional database. There is a different representation data with sequential database that will be used as an input for FreeSpan method. Therefore, we build sequential database from medical databases. The generated patterns are used as knowledge to predict the sequential disease. Therefore, the medical representative can take preventive and curative actions more precise. To make the sequential disease patterns easily to understand, we visualize the pattern using graph.**

I. INTRODUCTION

DATA can be interpreted from a temporal or a sequential perspective, i.e. the order appearance elements are relevant. There are several methods to discover sequential patterns along the data currently, e.g. AprioriAll [1], GSP [2], FreeSpan [3], PrefixSpan [4], SPADE [5], CloSpan [6]. Several studies have contributed to the efficient mining of sequential patterns or other frequent patterns in time related data.

Sequential pattern mining, which discovers frequent sequences as patterns in a database, is an important data mining research problem with broad applications. In this paper, we employ FreeSpan [3] for discovering sequential disease pattern in the medial databases. FreeSpan is employed because the method is interesting, scalable, and efficient. Sequence disease represents an important source of potentially new medical knowledge to predict the sequential disease. Based on the knowledge, the medical representatives can take preventive and curative actions more precise.

The remaining of the paper is organized as follows. In Section 2 we overview the FreeSpan. The demonstrations to discover sequential pattern disease using FreeSpan are presented in Section 3. We summarize our work in Section 4.

Manuscript received September 27, 2009. This work was supported by the Direktorat Jenderal Pendidikan Tinggi, Departemen Pendidikan Nasional under Grant 110/SP2H/PP/DP2M/IV/2009.

Oviliani Yenty Yuliana is a lecturer at Informatics Engineering Department, Petra Christian University, Surabaya 60236 Indonesia (corresponding author to provide phone: +62-31-2983455; e-mail: ovi@petra.ac.id).

Silvia Rostianingsih is a lecturer at Informatics Engineering Department, Petra Christian University, Surabaya 60236 Indonesia (corresponding author to provide phone: +62-31-2983455; e-mail: silvia@petra.ac.id).

Gregorius Satia Budhi is a lecturer at Informatics Engineering Department, Petra Christian University, Surabaya 60236 Indonesia (corresponding author to provide phone: +62-31-2983455; e-mail: greg@petra.ac.id).

II. FREESPAN

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. An itemset is a subset of items. A sequence is an ordered list of itemsets. A sequence s is denoted by $<s_1\ s_2\ \ldots\ s_l>$, where s_j is an itemset, i.e., $s_j \subseteq I$ for $1 \le j \le l$. s_j is also called an element of the sequence, and denoted as $(x_1\ x_2\ \ldots\ x_m)$, where x_k is an item, i.e., $x_k \in I$ for $1 \le k \le m$. For brevity, the brackets are omitted if an element has only one item. That is, element (x) is written as x. An item can occur at most once in an element of a sequence, but can occur multiple times in different elements of a sequence. A sequence $\alpha = <a_1\ a_2\ \ldots\ a_n>$ is called a subsequence of another sequence $\beta = <b_1\ b_2\ \ldots\ b_m>$ and β a super sequence of α, denote as $\alpha \subseteq \beta$, if there exist integer $1 \le j_1 < j_2 < \ldots < j_n \le m$ such that $a_1 \subseteq b_{j_1}$, $a_2 \subseteq b_{j_2}$, …, $a_n \subseteq b_{j_n}$.

A sequence database S is a set of tuples $<sid, s>$, where *sid* is a sequence identifier and s is a sequence. A tuple $<sid, s>$ in a sequence database S is said to contain a sequence α, if α is a subsequence of s, i.e., $\alpha \subseteq s$. The support of a sequence α in a sequence database S is the number of tuples in the database containing α. Given a positive integer ξ as the support threshold, a sequence α is called a sequential pattern in sequence database S if the sequence is contained by at least ξ tuples in the database.

Given a sequence database S and the support threshold ξ, FreeSpan algorithm mine the complete set of sequential patterns as follows. At last, visualize the sequential diseases using graph.

A. Generate a Length-1 Sequential Patterns

Scan *S* once to find the set of frequent items on distinct item of tuples in *S*. Save and sort them by descending order in a frequent list. The frequent list will be used throughout the mining process. The frequent items which fulfill the support threshold are used to construct a frequent item matrix *F*.

B. Construct a Frequent Item Matrix

Construct a frequent item matrix *F* by scanning the database to count the occurrence frequency of each length-2 sequence formed by items in the frequent list, as follows. For frequent list $< i_1, i_2, \ldots, i_n >$, *F* is a triangular matrix F[j, k], where $1 \le j \le m$ and $1 \le k \le j$, m is the number of frequent items. F[j, j] (for $1 \le j \le m$) has only one counter (recording the appearance of sequence <jj>), whereas every other slot F[j, k] ($1 \le j \le m$ and $1 \le k \le j$) has three counters: (*A*, *B*, *C*). *A* is the number of occurrences that i_k occurs after i_j, i.e. the sequence contains $<i_j\ i_k>$. *B* is that i_k occurs before i_j, i.e. the sequence contains $<i_k\ i_j>$. C is that i_k occurs concurrently with i_j, i.e. the sequence contains $<(i_j\ i_k)>$.

C. Generate a Length-2 Sequential Patterns

Generate a length-2 sequential patterns and a set of projected database from the frequent item matrix. For each counter, if the value in the counter is no less than minimum support, output the corresponding frequent pattern.

1) Generate annotations on item-repeating patterns for row j: For the diagonal slot, if *F*[j, j] ≥ minimum support, generate $<jj^+>$. The count of <jjj>, <jjjj>, … should be registered in the next round. For a column i ≠ j, follow rules: If *F*[i, i] ≥ minimum support, the annotation should contain i^+. It means there are potentially more than one I appearing in the sequential pattern. If *F*[j, j] ≥ minimum support, the annotation should contain j^+. Moreover, if only one of the three counter of *F*[i, j] is frequent, sequence is used as the annotation; otherwise set is used. It means, enhance string filtering. The annotation of an item repeating pattern is of the form $\$a_i^{\Upsilon}a_i^{\Upsilon}\$$, where \$...\$ can be either <…> (indicating looking for a particular ordered sequence only) or {…} (indicating looking for any ordered sequence), and a_i^{Υ} can be either a_i^+ (indicating looking for more than one occurrence of a_i), or a_i (i.e. no repeating a_i's) – notice that at least one of a_i and a_j is a repeating one.

2) Generate annotations on projected databases for row j: for each i < j, if *F*[i, j], *F*[k, j], and *F*[i, k] (k < i) may form a pattern generating triple (i.e. all the corresponding pairs are frequent), k should be added to i's projected column set. After examining all columns in front of i, the set of projected columns is determined. Output the annotation containing I, j, and the set of projected columns. If there is a choice between sequence or set, sequence is preferred since it enforces a stronger restriction in the projection. The annotation of the projected database is of the form $\$a_ia_j\$: \{b_p, \ldots, b_q\}$, where \$...\$ has the same convention as the item repeating pattern, and $\{b_p, \ldots, b_q\}$ represents a set of frequent items which may occur together with $\$a_ia_j\$$ to form longer sequential patterns in subsequent mining.

D. Generate a Length-3 Sequential Patterns

Based on the annotation generated from the matrix, scan the database one more time and generate the item-repeating patterns and projected databases. The remaining mining will be confined to each small projected database, by examining only the corresponding patterns enclosed in the header set.

III. Discovering Sequential Disease Patterns

In our study, we use RSU Dr. Soetomo medical databases (OLTP). Partly diagnosis structure table is shown in Table 1. The disease codes base on an international standard, ICD-X [7]. For example, A09.X is a disease code for Diarrhoea and gastroenteritis of presumed infectious origin. We built sequential database (OLAP) from diagnosis table as shown in Table 2. Given the sequence disease database *S*, the first two columns, in Table 2. The minimum support is 2. The discovering sequential disease patterns process can be done. The mining conceptual framework is show in Figure 1.

TABLE I
PARTLY DIAGNOSIS STRUCTURE TABLE

Patient Id	Patient In	Diagnosis 1	Diagnosis 2	Diagnosis 3
1450	04/14/07	A09.X	A41.9	
1450	12/23/07	E86.X		
1450	05/25/08	A09.X		
1450	03/07/09	I10.X	E86.X	
…				
1456	08/09/07	C34.9	J90.X	
1456	0/31/08	C34.9	D63.0	J90.X

The first step (section 2.1), generate a length-1 sequential patterns, i.e. frequent list, by scanning *S*. The frequent list is sorted in descending order, i.e. <A09.X: 6, E86.X: 5, A41.9: 4, C34.9: 4, D63.0: 3, I10.X: 3, J90.X: 2>. E14.9 disease is not included into frequent list because its support less than 2.

TABLE II
A DISEASE SEQUENCE DATABASE

Patient Id	Sequence	Disease Pattern
1450	<(A09.X A41.9) E86.X A09.X (I10.X E86.X)>	{A09.X, A41.9, E86.X, I10.X}
1451	<(A09.X D63.0) (E86.X C34.9) A09.X (D63.0 E14.9)>	{A09.X, C34.9, D63.0, E14.9, E86.X}
1452	<(I10.X J90.X) (A09.X D63.0) I10.X A09.X D63.0>	{A09.X, D63.0, I10.X, J90.X}
1453	<(A09.X C34.9) (E86.X C34.9) A41.9>	{A09.X, E86.X, A41.9, C34.9}
1454	<I10.X (A09.X A41.9) A09.X E86.X A09.X (I10.X A41.9 C34.9)>	{A09.X, A41.9, C34.9, E86.X, I10.X}
1455	<A09.X (A09.X E86.X) (A09.X A41.9)>	{A09.X, A41.9, E86.X}
1456	<(C34.9 J90.X) (C34.9 D63.0 J90.X)>	{C34.9, D63.0, J90.X}

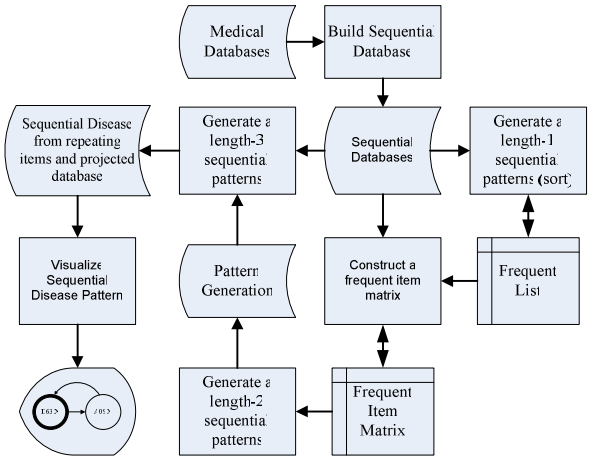


Fig. 1. The mining conceptual framework

The second step (section 2.2), construct a frequent item matrix. In our example, the frequent list consists of 7 items. It leads to generate a 7 X 7 triangular frequent item matrix, with every counter initialized to 0. For example, we fill up the counter matrix with the first tuples sequence disease in Table 2, i.e. <(A09.X A41.9) E86.X A09.X (I10.X E86.X)>. It generates a combination sequence diseases of <(A09.X A41.9)>, <(I10.X E86.X)>, <A09.X A09.X>, <A09.X E86.X>, <A09.X I10.X>, <A41.9 A09.X>, <A41.9 E86.X>, <A41.9 I10.X>, <E86.X A09.X>, <E86.X E86.X>, and <E86.X I10.X>. The sequence <A09.X I10.X> increases the counter matrix *F*[A09.X, I10.X] by 1, i.e. *F*[A09.X, I10.X] = (1,0,0). The <A41.9 A09.X> and <(A09.X A41.9)> increase the counter matrix *F*[A09.X, A41.9] by 1, i.e. *F*[A09.X, A41.9] = (0,1,1). The <A09.X E86.X> and <E86.X A09.X> increase the counter matrix *F*[A09.X, E86.X] by 1, i.e. *F*[A09.X, E86.X] = (1,1,0). The <E86.X I10.X> and <(I10.X E86.X)> increase the counter matrix *F*[E86.X, I10.X] by 1, i.e. *F*[E86.X, I10.X] = (1,0,1). The <A41.9, E86.X> increases the counter matrix *F*[E86.X, A41.9] by 1, i.e. *F*[E86.X, A41.9] = (0,1,0). The <A41.9 I10.X> increases the counter matrix *F*[A41.9, I10.X] by 1, i.e. *F*[A41.9, I10.X] = (1,0,0). The last <A09.X A09.X> increases the counter matrix *F*[A09.X, A09.X] by 1 (i.e. *F*[A09.X, A09.X] = 1) and <E86.X E86.X> increases the counter matrix *F*[E86.X, E86.X] by 1 (i.e. *F*[E86.X, E86.X] = 1). The process continues along tuples the sequential database. At the end of the second step, the frequent item matrix is shown in Figure 2. The counter matrixes which are greater than or equal minimum support are written in a bold font.

A09.X	**5**						
E86.X	(**5**,**4**,1)	1					
A41.9	(**3**,**2**,**3**)	(**3**,**2**,0)	1				
C34.9	(**3**,1,1)	(1,1,**2**)	(1,1,1)	**2**			
D63.0	(**2**,**2**,**2**)	(1,1,0)	(0,0,0)	(**2**,1,1)	**2**		
I10.X	(**3**,**2**,0)	(**2**,1,1)	(**2**,1,1)	(0,1,1)	(1,1,0)	**2**	
J90.X	(0,1,0)	(0,0,0)	(0,0,0)	(1,1,1)	(0,**2**,1)	(0,1,1)	1
	A09.X	E86.X	A41.9	C34.9	D63.0	I10.X	J90.X

Fig. 2. The frequent diseases matrix

The third step (section 2.3), generate a length-2 sequential patterns, annotations on item repeating patterns, and annotations on the projected databases as shown in Table 3. This step is based on the frequent item matrix F, Figure 2. For instance, we demonstrate the process for two rows I10.X and D63.0.

TABLE III
PATTERN GENERATION FROM THE FREQUENT DISEASE MATRIX

Disease	Output Length-2 Sequential Patterns	Ann. on Repeating Items	Ann. on Projected DBs
J90.X	<J90.X D63.0>: 2	<J90.X D63.0$^+$>	∅
I10.X	<A09.X I10.X>: 3, <I10.X A09.X>: 2, <E86.X I10.X>: 2, <A41.9 I10.X>: 2, <I10.X I10.X>: 2	{A09.X$^+$ I10.X$^+$}, <E86.X I10.X$^+$>, <A41.9 I10.X$^+$>, <I10.X I10.X$^+$>	<E86.X I10.X>: {A09.X}, <A41.9 I10.x>: {A09.X E86.X}
D63.0	<A09.X D63.0>: 2, <D63.0 A09.X>: 2, <(A09.X D63.0)>: 2, <C34.9 D63.0>: 2, <D63.0 D63.0>: 2	{A09.X$^+$ D63.0$^+$}, <C34.9$^+$ D63.0$^+$>, <D63.0 D63.0$^+$>	∅
C34.9	<A09.X C34.9>: 3, <(E86.X C34.9)>: 2, <C34.9 C34.9>: 2	<A09.X$^+$ C34.9$^+$>, <E86.X C34.9$^+$>, <C34.9 C34.9$^+$>	<(E86.X C34.9)>: {A09.X}
A41.9	<A09.X A41.9>: 3, <A41.9 A09.X>: 2, <(A09.X A41.9)>: 3, <E86.X A41.9>: 3, <A41.9 E86.X>: 2	{A09.X$^+$ A41.9}	<E86.X A41.9>: {A09.X}
E86.X	<A09.X E86.X>: 5, <E86.X A09.X>: 4	{A09.X$^+$ E86.X}	∅
A09.X	<A09.X A09.X>: 5	<A09.X A09.X$^+$>	∅

The I10.X row has 5 frequent counters matrix respectively as follows. *F*[A09.X, I10.X] = (3,2,0)

generates two length-2 sequential patterns <A09.X I10.X>: 3 and <I10.X A09.X>: 2. *F*[E86.X, I10.X] = (2,1,1) generates a length-2 sequential pattern <E86.X I10.X>: 2. *F*[A41.9, I10.X] = (2,1,1) generates a length-2 sequential pattern <A41.9 I10.X>: 2. At last, *F*[I10.X, I10.X] = 2 generates a length-2 sequential pattern <I10.X I10.X>: 2. Since both *F*[A09.X, A09.X] and *F*[I10.X, I10.X] are frequent, the annotation on repeating items {A09.X$^+$ I10.X$^+$} is generated which means one need to examine multiple occurrences of A09.X's dan I10.X's and their combinations in the next scan. In addition, *F*[I10.X, I10.X] are frequent, add <I10.X I10.X$^+$> to the annotation on repeating items. Moreover, there are still two frequent items in the I10.X row, i.e. E86.X and A41.9 columns. Each column has only one frequent counter with *F*[I10.X, I10.X] is frequent. *F*[E86.X, E86.X] and *F*[A41.9, A41.9], however, are not frequent. Therefore the annotation <E86.X I10.X$^+$> and <A41.9 I10.X$^+$> are generated. Furthermore, since *F*[A41.9, I10.X], *F*[E86.X, I10.X], and *F*[E86.X, A41.9] form a pattern generating triple, and *F*[A41.9, I10.X] = (2,1,1) which means only <A41.9 I10.X> is valid. The annotation for the projected database should be <A41.9 I10.X>: {A09.X, E86.X}. It indicates generating <A41.9 I10.X> projected database with item {A09.X, E86.X} included. *F*[E86.X, I10.X], *F*[A09.X, I10.X], and *F*[A09.X, E86.X] also form a pattern generating triple. *F*[E86.X, I10.X] = (2,1,1), it means only <E86.X I10.X> is valid. The annotation for the projected database should be <E86.X I10.X>: {A09.X}. It indicates generating <E86.X I10.X> projected database with only item {A09.X} included. As a result see Table 3, the second row, i.e. disease I10.X.

The D63.0 row also has 5 counter frequent, which leads to generate 5 length-2 sequential pattern: <A09.X D63.0>: 2, <D63.0 A09.X>: 2, <(A09.X D63.0)>: 2, <C34.9 D63.0>: 2, <D63.0 D63.0>: 2. *F*[A09.X, A09.X], *F*[D63.0, D63.0], and *F*[C34.9, C34.9] are frequent. Since more than one counter frequent in *F*[A09.X, D63.0] = (2,2,2), the annotation on repeating items {A09.X$^+$ D63.0$^+$} is generated. Since only one counter frequent in *F*[C34.9, D63.0] = (2,1,1), the annotation on repeating items <C34.9$^+$ D63.0$^+$> is generated. In addition, the annotation on repeating items <D63.0 D63.0$^+$> is generated. Furthermore, since *F*[C34.9, D63.0], *F*[A41.9, D63.0], and *F*[A41.9, C34.9] do not form a pattern generating triple. It means no other item could be co-frequented with <C34.9, D63.0>, there is no projected database annotation with A41.9. As a result see Table 3, the thrid row, i.e. disease D63.0.

The fourth step (section 2.4), generate a length-3 sequential patterns based on pattern generation in Table 3, i.e. projecting database and generating repeating pattern. There are 5 annotation on projected databases in Table 3 the fourth column, i.e. <E86.X I10.X>: {A09.X}, <A41.9 I10.x>: {A09.X E86.X}, <(E86.X C34.9)>: {A09.X}, and <E86.X A41.9>: {A09.X}. For instance, we demonstrate the annotation <E86.X I10.X>: {A09.X} by scanning sequential database one or more times. As a result, the projected databases are <A09.X E86.X A09.X (I10.X E86.X)> and <I10.X A09.X A09.X E86.X A09.X I10.X>. Based on the projected database, sequential patterns <A09.X E86.X A09.X I10.X>:2, <E86.X A09.X I10.X>:2, and <A09.X E86.X I10.X>:2 are discovered. The generated length-3 sequential patterns by projected databases are shown in Table 4.

TABLE IV
SEQUENTIAL DISEASE PATTERNS FROM PROJECTED DATABASE

Ann. on Projected DBs	Projected Database	Sequential Patterns
<E86.X I10.X>: {A09.X}	<A09.X E86.X A09.X (I10.X E86.X)>, <I10.X A09.X A09.X E86.X A09.X I10.X>	<A09.X E86.X A09.X I10.X>:2 <E86.X A09.X I10.X>:2 <A09.X E86.X I10.X>:2 <(A09.X A41.9) I10.X>:2
<A41.9 I10.x>: {A09.X E86.X}	<(A09.X A41.9) E86.X A09.X (I10.X E86.X)>, <(A09.X A41.9) A09.X E86.X A09.X (I10.X A41.9)>	<A41.9 E86.X A09.X I10.X>:2 <A41.9 E86.X I10.X>:2 <A41.9 A09.X I10.X>:2
<(E86.X C34.9)>: {A09.X}	<A09.X (E86.X C34.9) A09.X>, <A09.X (E86.X C34.9)>	<A09.X (E86.X C34.9)>:2
<E86.X A41.9>: {A09.X}	<A09.X E86.X A41.9>, <A09.X A09.X E86.X A09.X A41.9>, <A09.X (A09.X E86.X) (A09.X A41.9)>	<A09.X E86.X A41.9>:3

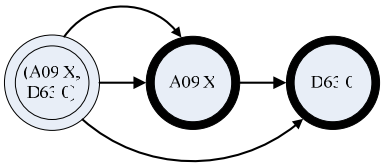
There are 14 annotations on repeating items including a looking for a particular and any ordered sequences (see Table 3 the third column). The particular ordered sequences are <J90.X D63.0$^+$>, <E86.X I10.X$^+$>, <A41.9 I10.X$^+$>, <I10.X I10.X$^+$>, <C34.9$^+$ D63.0$^+$>, <D63.0 D63.0$^+$>, <A09.X$^+$ C34.9$^+$>, <E86.X C34.9$^+$>, <C34.9 C34.9$^+$>, and <A09.X A09.X$^+$>. They should not be generated for further repeating patterns. The particular ordered sequences use the output length-2 sequential patterns. Moreover, there are 4 any ordered sequences, i.e. {A09.X$^+$, I10.X$^+$}, {A09.X$^+$, D63.0$^+$}, {A09.X$^+$, A41.9}, and {A09.X$^+$, E86.X}. For example, we demonstrate how to generate repeating items for {A09.X$^+$, I10.X$^+$}. Generate several sequence from {A09.X$^+$, I10.X$^+$}, e.g. <A09.X, I10.X, I10.X>, <I10.X, A09.X, I10.X>, <A09.X, I10.X, I10.X>. For each sequence, scan sequential databases to discover

the frequent sequence pattern. For our example, there are 3 frequent sequence patterns, i.e. <A09.X, A09.X, I10.X>: 2, <I10.X, A09.X, I10.X>: 2, and <I10.X, A09.X, A09.X>: 2. The generated length-3 sequential patterns by repeating item are shown in Table 5.

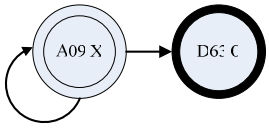
TABLE V
SEQUENTIAL DISEASE PATTERNS FROM REPEATING ITEMS

Ann. on Repeating items	Sequential Patterns
{A09.X$^+$, I10.X$^+$}	<A09.X, A09.X, I10.X>: 2
	<I10.X, A09.X, I10.X>: 2
	<I10.X, A09.X, A09.X>: 2
{A09.X$^+$, D63.0$^+$}	<(A09.X, D63.0), A09.X, D63.0>: 2
	<(A09.X, D63.0), A09.X>: 2
	<(A09.X, D63.0), D63.0>: 2
	<A09.X, A09.X, D63.0>: 2
	<D63.0, A09.X, D63.0>: 2
{A09.X$^+$, A41.9}	<(A09.X, A41.9), A09.X>: 2
	<A09.X, A09.X, A41.9>: 2
{A09.X$^+$, E86.X}	<A09.X, E86.X, A09.X>: 4
	<A09.X, A09.X, E86.X>: 2
{A09.X, A09.X$^+$}	<A09.X, A09.X, A09.X>: 2

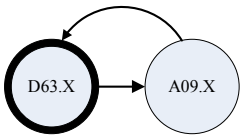
The last step as an optional step, visualize a sequential disease pattern using graph. For instance an annotation on repeating items {A09.X$^+$, D63.0$^+$} in Table 5 is visualized in Figure 3. Double circle is used for start sequence, bold circle is used for end sequence, and normal circle is used for transition sequence.



A. <(A09.X, D63.0), A09.X, D63.0>
<(A09.X, D63.0), A09.X>
<(A09.X, D63.0), D63.0>



B. <A09.X, A09.X, D63.0>



C. <D63.0, A09.X, D63.0>

Fig. 3. An {A09.X$^+$, D63.0$^+$} sequential pattern graph

IV. CONCLUSIONS

FreeSpan method can be used for mining sequential diseases pattern form database sequential. In case medical database was built in different representative data, i.e. transactional database. The medical database should be transformed into a sequential database first. Then use FreeSpan to mine sequential diseases pattern. The generated patterns can be used a knowledge to predict sequential diseases. As a result the medical representative can take preventive and curative action more precisely.

ACKNOWLEDGMENT

This work was supported by the Direktorat Jendral Pendidikan Tinggi, Departemen Pendidikan Nasional under Grant 110/SP2H/PP/DP2M/IV/2009.

REFERENCES

[1] R. Agrawal and R. Srikant, "Mining Sequential Patterns", *The eleventh international conference on data engineering*, IEEE Computer Society Press, Taipei Taiwan, March 6-10 1995, pp. 3-14.

[2] Srikant, R. and R. Agrawal, *Mining Sequential Patterns: Generalizations and Performance Improvments*, IBM Research Division, San Jose CA, 1996.

[3] J. Han, J. Pei, B.M. Asl, Q. Chen, U. Dayal, and M.C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining", *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*, Boston MA USA, August 20-23 2000, pp. 355-359.

[4] J. Pei, J. Han, B.M. Asl, Q. Chen, U. Dayal, and M.C. Hsu, "Prefixspan: Mining Sequential Patterns Efficiently by Prefix Projection Pattern Growth", Proceedings of the 17th International Conference on Data Engineering, 2001, pp. 215-224.

[5] M. Zaki, "Spade: An Efficient Algorithm for Mining Frequent Sequences", *Machine Learning Journal, special issue on Unsupervised Learning*, Kluwer Academic Publishers, Boston, 2000.

[6] X. Yan, J. Han, and R. Afshar, "Clospan: Mining Closed Sequential Patterns in Large Datasets", *Proceeding of SIAM International Conference on Data Mining*, 2003, pp. 438-457.

[7] World Health Organization, ICD-10 Version 2007. Available: http://apps.who.int/classifications/apps/icd/icd10online